Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones

Matthew French, Scott Smith, and Caroline Nguyen

Presentation Outline

1. Introduction and Past Work
2. Manual Model Generation
3. Automatic Model Generation
4. Analysis of Models
5. Results and Limitations

Presentation Outline

1. Introduction and Past Work
2. Manual Model Generation
3. Automatic Model Generation
4. Analysis of Models
5. Results and Limitations

Introduction

- Power usage is an important metric to measure on smartphones.
- Smartphone app developers want to create powerful and feature-rich applications.
- Smartphone users want their batteries to last a long time.
- Smartphone app developers cannot easily measure how much power their application is drawing.
  - This leads to unnecessarily power-hungry apps.

Past Work in Power Modeling

- Some past work relied on in-depth knowledge of the processing unit.
  - R. Joseph and M. Martonosi, “Run-time power estimation in high-performance microprocessors,” in Proc. Int. Symp. Low Power Electronics & Design, Aug. 2001, pp. 135–140.
  - C. Isci and M. Martonosi, “Runtime power monitoring in high-end processors: Methodology and empirical data,” in Proc. Int. Symp. Microarchitecture, Dec. 2003, pp. 93–104.
- Other past work treats processors as black boxes.
  - F. Bellosa, “The benefits of event-driven energy accounting in power-sensitive systems,” in Proc. Special Interest Group on Operating Systems European Wkshp., 2006, pp. 37–42.
  - G. Contreras, et al., “XTREM: a power simulator for the Intel XScale,” in Proc. Conf. Languages, Compilers, and Tools for Embedded Systems, June 2004, pp. 115–125.
- Both **only** analyze power consumption of the processor

Past Work in Power Modeling

- Other work created component-based power models
  - T. Cignetti, K. Komarov, and C. Ellis, “Energy estimation tools for the Palm,” in Proc. of the ACM Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2000, pp. 96–103.
  - A. Shye, B. Scholbrock, and G. Memik, “Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures,” in Proc. Int. Symp. Microarchitecture, 2009, pp. 168–178.
  - J. Flinn, M. Satyanarayanan, Powerscope: a tool for profiling the energy usage of mobile applications, in: Mobile Computing Systems and Applications, 1999, Proceedings, WMCSA’99, Second IEEE Workshop on, pp. 2–10.
- External power measurement equipment was required
- Both models correlated visible operating system state variables with power consumption while running a series of standard applications
  - Problematic when training applications don't fully utilize all hardware
  - Better to explicitly test all system states

Novel Work in this Paper

- Manually generated power models for two phones: HTC Dream and HTC Magic
  - Models consider power draw due to CPU, LCD, GPS, Wi-FI, cellular, and audio-components
    - First time a smartphone GPS power model had been described
    - Power consumption between different components appears independent
- Variation in power consumption between phones is measured and analyzed
  - Variation between phones of the same model → low variation
  - Variation between phones of different models → large variation
- Creation of an automated power model construction technique
  - Doesn’t need additional hardware to be attached to the phone
  - Many phones now have current sensors, so this is less useful on newer devices

Presentation Outline

1. Introduction and Past Work
2. Manual Model Generation
3. Automatic Model Generation
4. Analysis of Models
5. Results and Limitations

Component-Level Power Modeling

Component state variables determine power model parameters:

- Wi-Fi, 3G states
- GPS modes
- CPU frequency/utilization
- LCD brightness
- Audio volume

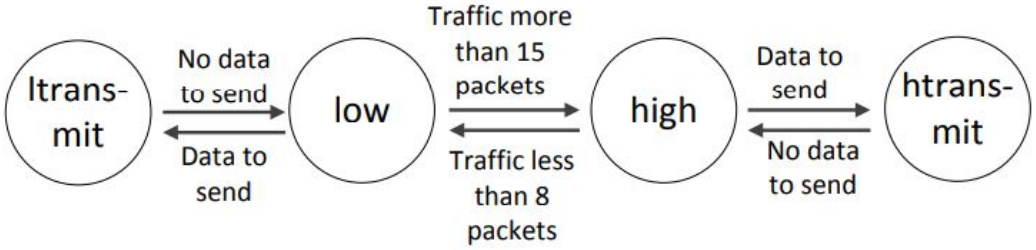


Figure 3: Wi-Fi interface power states.

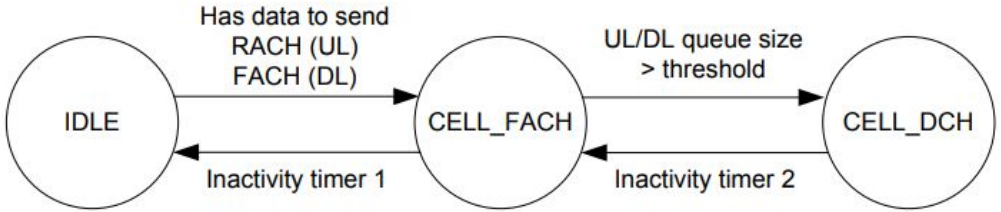


Figure 4: 3G interface power states.

Model Generation

Setup:

1. Monsoon FTA22D meter provides constant voltage and measures power usage
2. Two programs to exercise all power states

For all smartphone components:

1. Set all other components into low-power states
2. Vary activity state for component to extreme values (ex. CPU utilization to low and high values, GPS to few and many connected satellites)
3. Determine if component has significant effect on power (exclude components that have an insignificant effect - ex. SD card)
4. Repeat experiments to determine model form for each component
   - LCD → β_{br} × brightness
   - GPS → β_{Gon} × GPS_on + β_{Gsl} × GPS_sl

Assumption

“...we found that the maximum error resulting from assuming that individual components are independent is 6.27%. This suggests that a sum of independent component-specific power estimates is sufficient to estimate system power consumption.”

This is not true in reality: the CPU might increase if the cellular interface is working hard. This assumption might be less true for modern phones.

The 6.27% number may not be accurate for modern phones.

Linear Regression

$$\begin{pmatrix} P_0 \\ P_1 \\ \dots \\ P_n \end{pmatrix} = \beta_1 \cdot \begin{pmatrix} U_{01} \\ U_{11} \\ \dots \\ U_{n1} \end{pmatrix} \dots + \beta_m \cdot \begin{pmatrix} U_{0m} \\ U_{1m} \\ \dots \\ U_{nm} \end{pmatrix} + c. \quad (3)$$

$U_{i,j}$ = System variable i in the jth state

β_i = Power coefficients of the linear regression

P_j = Power consumption when all system variables are in the jth state

C = Minimum system power consumption (constant)

Model

Table 2: HTC Dream Power Model

Model	$(\beta_{uh} \times freq_h + \beta_{ul} \times freq_l) \times util + \beta_{CPU} \times CPU\_on + \beta_{br} \times brightness + \beta_{Gon} \times GPS\_on + \beta_{Gsl} \times GPS\_sl + \beta_{Wi\text{-}Fi\_l} \times Wi\text{-}Fi_l + \beta_{Wi\text{-}Fi\_h} \times Wi\text{-}Fi_h + \beta_{3G\_idle} \times 3G_{idle} + \beta_{3G\_FACH} \times 3G_{FACH} + \beta_{3G\_DCH} \times 3G_{DCH}$						
Category	System variable	Range	Power coefficient	Category	System variable	Range	Power coefficient
CPU	util	1–100	β_{uh}: 4.34 β_{ul}: 3.42	LCD	brightness	0–255	β_{br}: 2.40
				GPS	GPS_on	0,1	β_{Gon}: 429.55
	$freq_l$,$freq_h$	0,1	n.a.		GPS_sl	0,1	β_{Gsl}: 173.55
	CPU_on	0,1	β_{CPU}: 121.46	Cellular	data_rate	0–∞	n.a.
Wi-Fi	npackets, R_{data}	0–∞	n.a.		downlink_queue	0–∞	n.a.
	$R_{channel}$	1–54	β_{cr}		uplink_queue	0–∞	n.a.
	$Wi\text{-}Fi_l$	0,1	$\beta_{Wi\text{-}Fi\_l}$: 20		$3G_{idle}$	0,1	β_{3G_idle}: 10
	$Wi\text{-}Fi_h$	0,1	$\beta_{Wi\text{-}Fi\_h}$: Equation 1		$3G_{FACH}$	0,1	β_{3G_FACH}: 401
Audio	Audio_on	0,1	β_{audio}: 384.62		$3G_{DCH}$	0,1	β_{3G_DCH}: 570

Model Consistency

- Intra-type variance exceeds 10.4% for no component (assuming power consumption difference distribution is Gaussian)
- Inter-type variance was much greater (up to 62% variation for cellular interfaces)

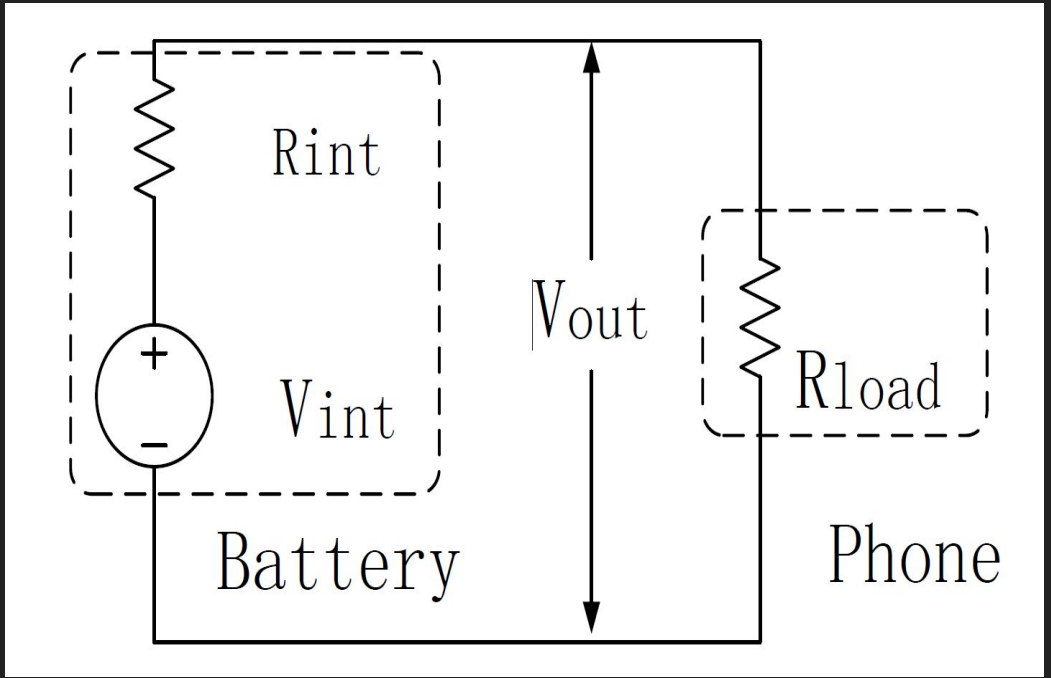
These results show that a model can be used if the phone was of the same type, but for phones of a different type, a new model must be created

Presentation Outline

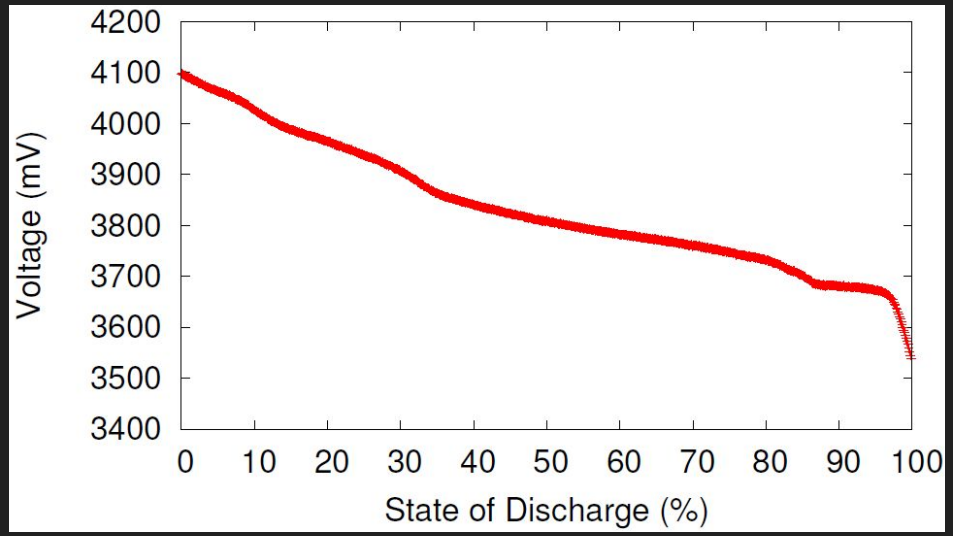
1. Introduction and Past Work
2. Manual Model Generation
3. Automatic Model Generation
4. Analysis of Models
5. Results and Limitations

Simplified Circuit Diagram for Lithium-Ion Batteries

- R_{int} and V_{int} are affected by temperature and discharge current
- Aging of the battery permanently alters R_{int} and V_{int}
- R_{load} changes depending on the power states and use of various internal components.



State of Discharge Curve



- Monotonically Decreasing
- Curve affected by: battery age, temperature, and discharge current

Using the State of Discharge Curve

$$P \times (t_1 - t_2) = E \times (SOD(V_1) - SOD(V_2))$$

- $[t_1, t_2]$ is the time interval
- P is average power consumption over time interval
- E is rated battery energy capacity
  - Deteriorates as the battery ages
- $SOD(V_i)$ is stage of discharge at Voltage V_i
  - This is a map from battery voltage to the percent of battery capacity remaining

Determining the State of Discharge Curve

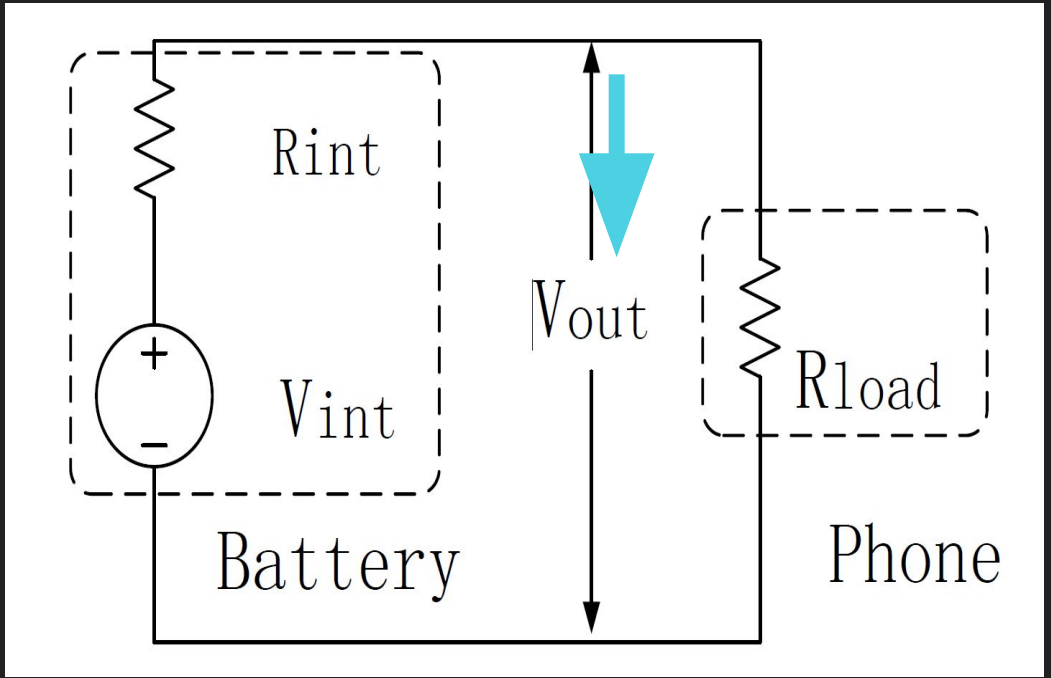
1. Charge battery to full
2. Put phone in a room temperature environment
    a. Reduces average variance caused by changing temperatures to around 4.3%
    b. More representative of normal use
3. Set the current draw to a fixed rate
    a. This presumably can be done by controlling which applications are running
    b. Fixing the current draw creates a linear relationship between SOD and time.
4. Record the voltage at fixed time intervals
5. Discharge the battery until empty
6. Create a piecewise linear function from Voltage to SOD based on the gathered points.

Determining the Energy Capacity

1. If your battery is new, read the label on the battery
2. If your battery is not new, you need to measure the energy capacity
    a. You can calculate the total energy based on the total power consumption and time it takes to discharge the battery.
    b. This requires you to know the power consumption of the connected components.
3. Impacted by discharge current, temperature, and battery age
    a. The effects of discharge current were tested by using the lowest and highest current draw rates possible during SOD curve generation.
    b. There was a less than 2.4% error in power consumption estimates in these worst case scenarios.

Measuring Battery Voltage

Need to minimize the current draw!



Steps for Automatic Model Generation

1. Obtain battery discharge curve for this phone
2. Determine power consumption for each state of each component
3. Perform regression to derive power model

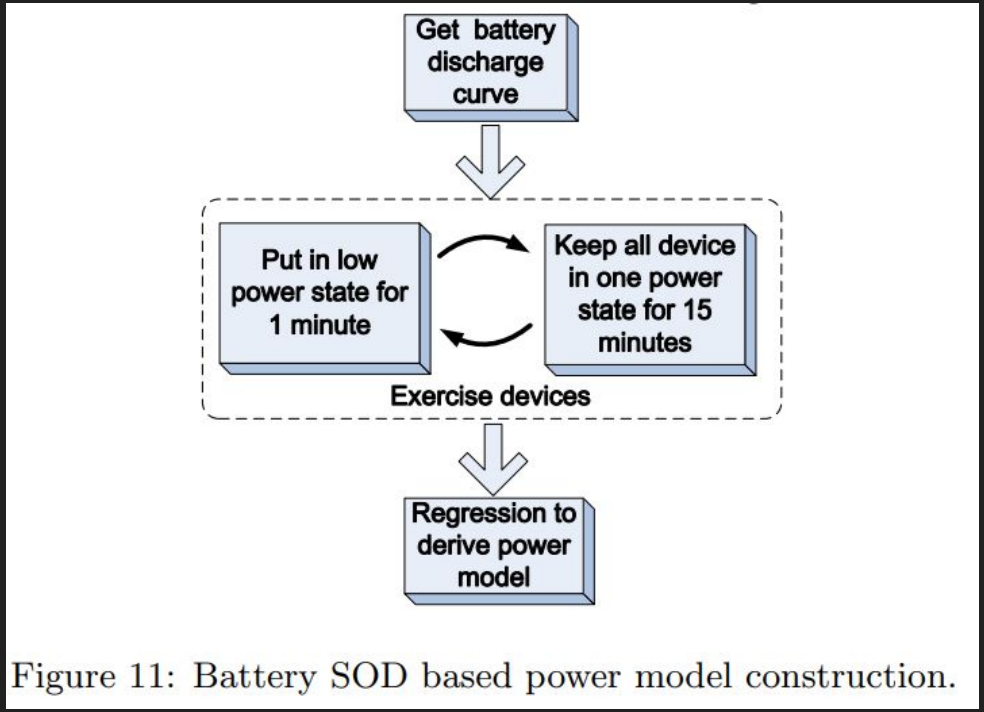


Figure 11: Battery SOD based power model construction.

Presentation Outline

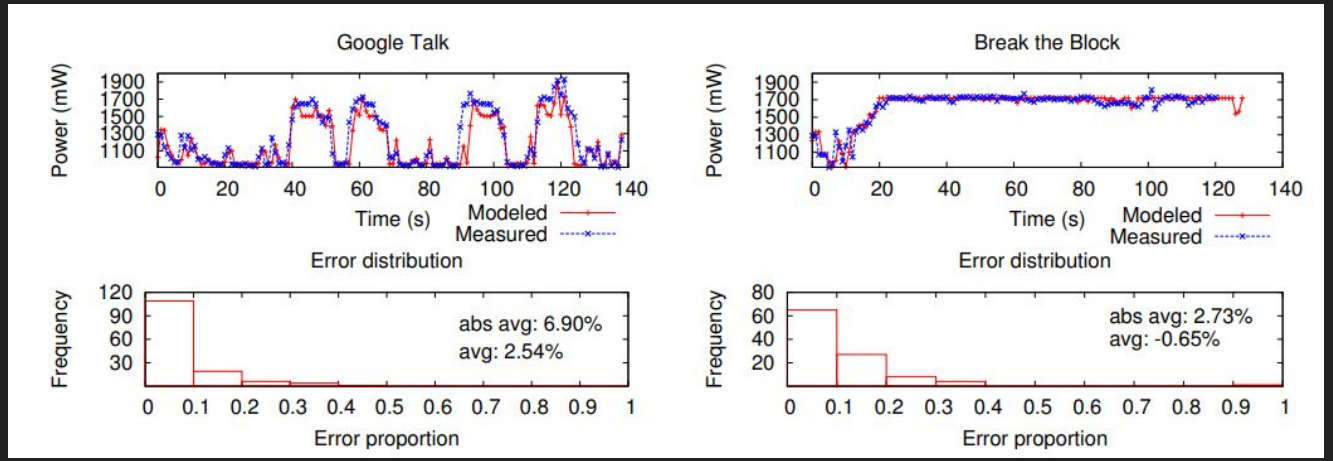
1. Introduction and Past Work
2. Manual Model Generation
3. Automatic Model Generation
4. Analysis of Models
5. Results and Limitations

6 Test Applications

- Break the Block: A game that uses CPU, LCD, and Audio.
- Google Talk: An instant message application that uses CPU, LCD, Wi-Fi/3G, and Audio.
- Google Maps: A web mapping application that uses CPU, LCD, Wi-Fi/3G, and GPS.
- The Weather Channel: A weather forecast application that uses CPU, LCD, Wi-Fi/3G, and GPS.
- YouTube: A web-based video sharing application that uses CPU, LCD, and Wi-Fi/3G.
- Browser: The default web browser on Android that uses CPU, LCD, and Wi-Fi/3G.

Accuracy of Model

- Long-term error (avg) is less than 2.5% over the application's lifespan
- Average error (abs avg) is less than 10% for 1 second intervals



PowerTutor/PowerBooter

- PowerTutor power estimation
  - Effects of application design and use on power consumption
  - 6 components
    - CPU, LCD, GPS, Wi-Fi, audio, and cellular interfaces
- PowerBooter
  - “Automatic battery state of discharge based power model generation technique”
  - Enables users to create power model construction without power meter

Presentation Outline

1. Introduction and Past Work
2. Manual Model Generation
3. Automatic Model Generation
4. Analysis of Models
5. Results and Limitations

Results

- PowerTutor
  - Accurate within 0.8%, at most 2.5% error (10-second intervals)
  - Power estimation app in Android App Market → 6000+ users
  - Used to compare power consumption of different apps
  - Estimates battery lifespan
- PowerBooter
  - Accurate within 4.1% of measured values (10-second intervals)
- Combined
  - Allows for power modeling/analysis among different smartphones and users

Possible Limitations

- Only tested with Android devices (iOS, Bada, etc. devices might work differently in a significant way)
- A lot of phones now have current meters
- This type of system does not apply to different or newer phones with different technology
  - Bluetooth
  - OLED vs LCD
  - State machines for Wi-Fi/Cellular might be different for some devices
  - Camera
- Some components might have higher relative draw on some devices than others
- Component independence assumption may not hold for all devices

Questions?